# 第 6 章 数据流机和归约机

传统的冯诺依曼型计算机总是顺序地执行指令，这是开发计算并行性的最大限制。

非冯诺依曼型计算机：

- 基于数据驱动的数据流计算机，使用数据流语言
- 基于驱动需求的归约计算机，使用函数式语言
- 基于模式匹配驱动的智能计算机，使用逻辑式语言
- 神经元计算机，仿照人脑组织结构和思维构成

# 6.1 数据流计算机

## 6.1.1 数据驱动的概念

- 数据驱动的数据流方式：只要一条或一组指令所要求的操作数全部准备就绪，就可立即激发相应的指令或指令组执行。执行结果的输出将送往等待这一数据的下一条或下一组指令。

如果其中一些指令因此而使所需用到的数据全部准备就绪，就可被激发执行。指令的执行基本上是无序的，完全受数据流的驱动，与指令在程序中出现的先后顺序无关。

## 控制驱动 VS. 数据驱动

### 控制驱动的控制流方式：

- 通过访问共享存储单元让数据在指令之间传递；
- 指令执行的顺序性隐含于控制流中，但却可以显示使用专门的控制操作符来实现并行处理；
- 指令执行的顺序受程序计数器控制，即控制令牌所支配。

### 数据驱动的数据流方式：

- 它没有通常的共享变量的概念，即没有共享存储数据的概念；
- 指令执行顺序只受指令中数据相关性的制约；
- 数据是以数据令牌方式直接在指令之间传递的。

数据令牌：实质上是一种表示某一操作数或参数已准备就绪的标志。一旦执行某一操作的所有操作数令牌都到齐，则标志着这一操作是什么操作，以及操作结果所得出的数据令牌应发送到哪些等待此数据令牌的操作的第几个操作数部件等有关信息，都将作为一个消息包(MessagePacket)，传送到处理单元或操作部件并予以执行。

只要数据不相关和资源可以利用，就可以并行，因而最有利于计算并行性的开发。

## 数据流的特性

①并行性：可同时并行执行多条指令（并行性通常是隐含的）.

②异步性：一旦一条指令所需求的数据令牌到达后，指令就可独立执行，而不必关心其他指令及数据情况如何.

③函数性：运算的执行都是局部操作，操作数是作为数据令牌直接传送的，每一组数据流操作都需要一组输入值，产生一组输出值.

④分散性：不需要控制执行次序，故不需要集中控制.

从语义上说，数据流是基于异步性和函数性的一种计算模型

# 6.1.2 数据流程序图和语言

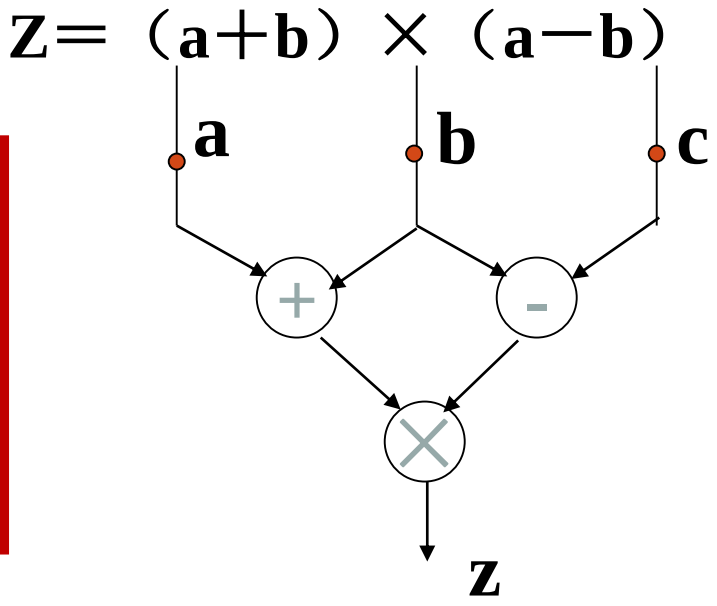
## 1、数据流程序图 数据流机器的机器语言

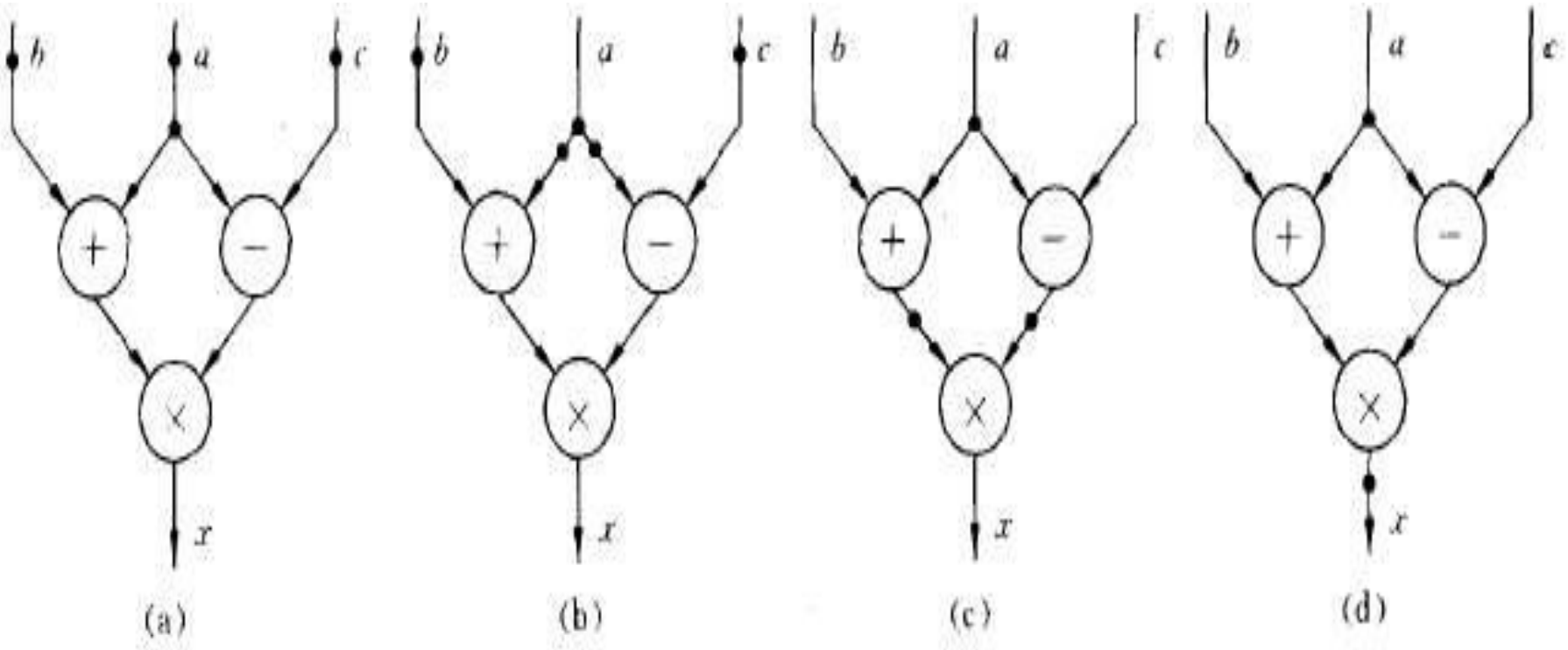
数据流程序图共有两种表示形式：有向图和活动片

### □ 有向图表示法

有向图由有限个结点集合与连接结点的单向分支线组成

*结点执行规则：*当一个结点的所有输入分支线上都出现数据令牌，且输出分支线上没有数据令牌时，该结点的操作即可执行。

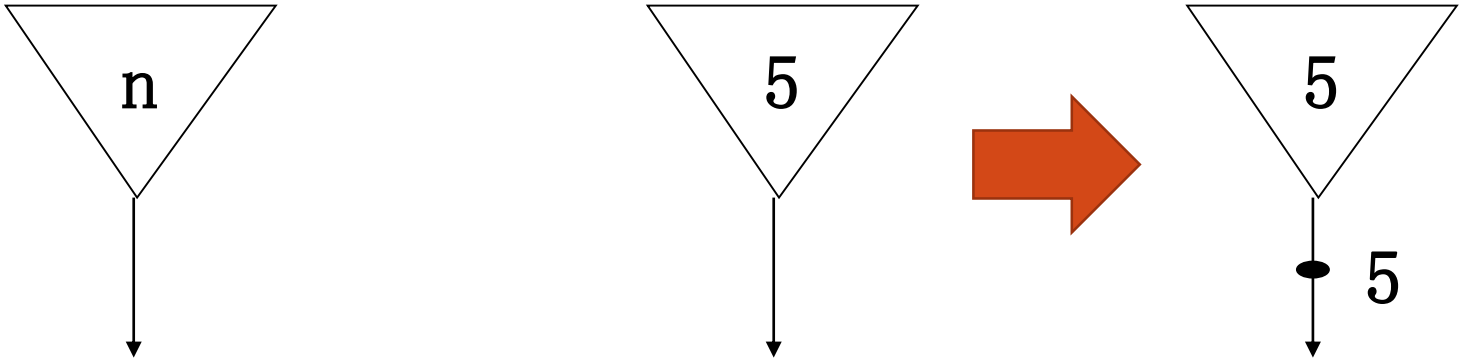





实心圆点代表数据令牌沿弧移动，在一段时间内描述数据在程序图中的流动状况；整个程序图中程序执行的过程是数据不断激发（驱动）的过程

## 常数发生器结点

用来产生一个常数，无输入端，激发后输出常数令牌



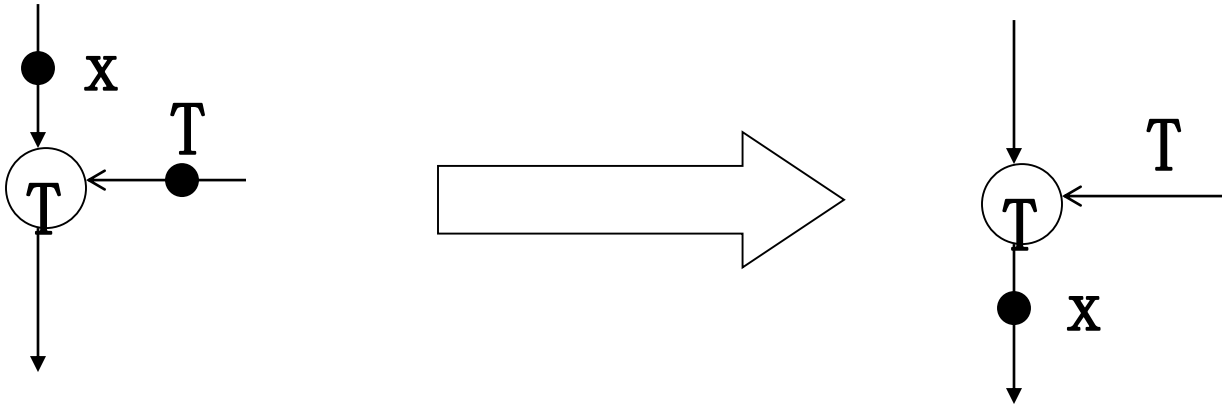

## 复制结点

该结点将一个令牌复制成两个相同的令牌，包括数据令牌和控制令牌。

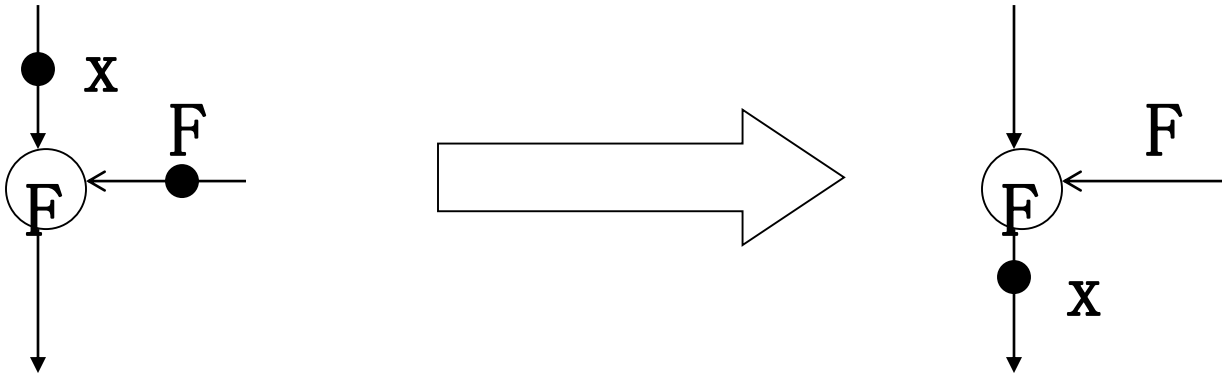

数据结点　　布尔结点

- 条件分支结点

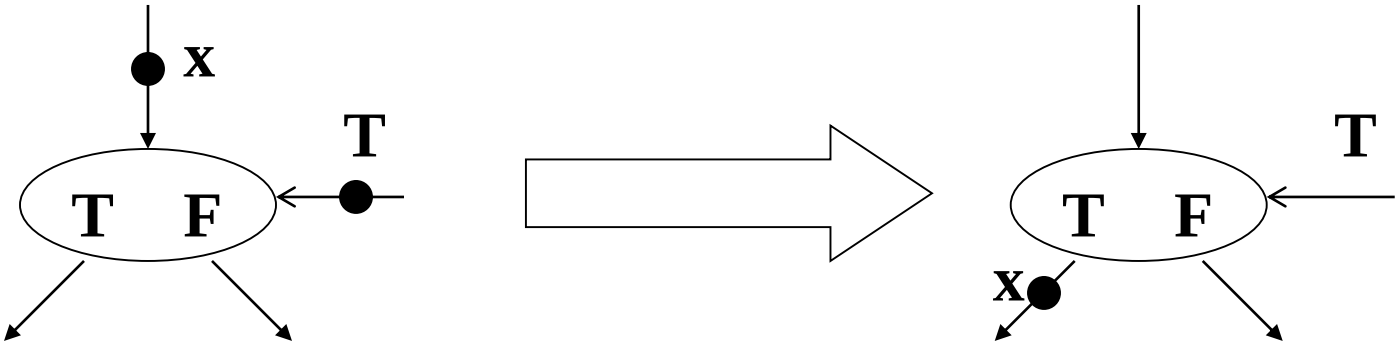
T门分支结点：



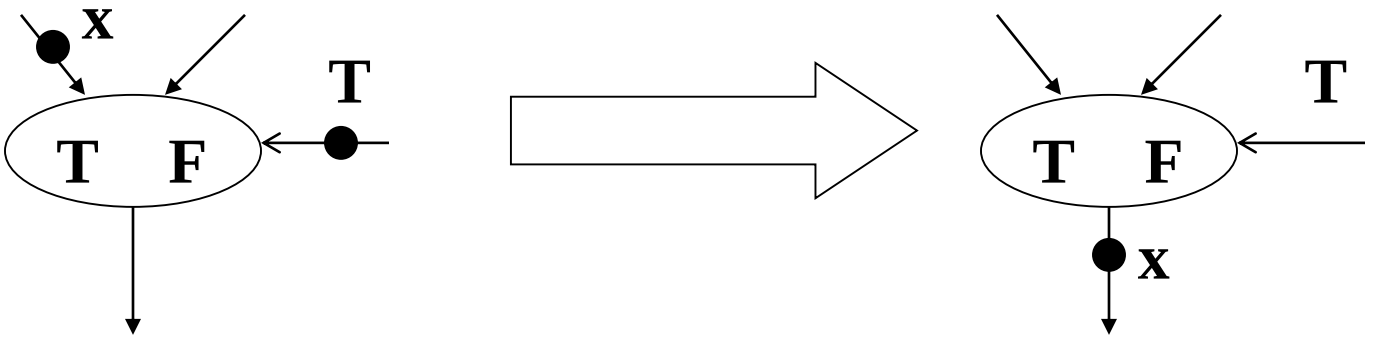

F门分支结点：



开关分支结点：

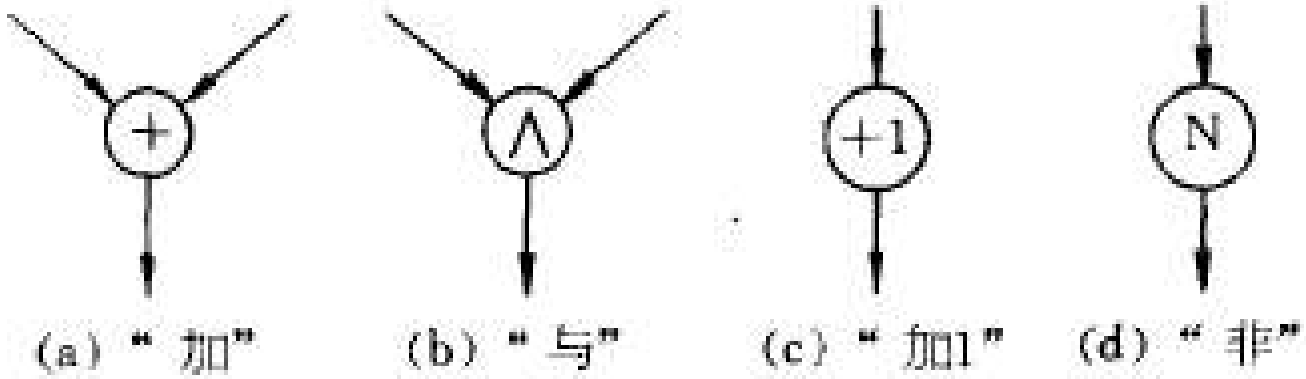



合并分支结点：



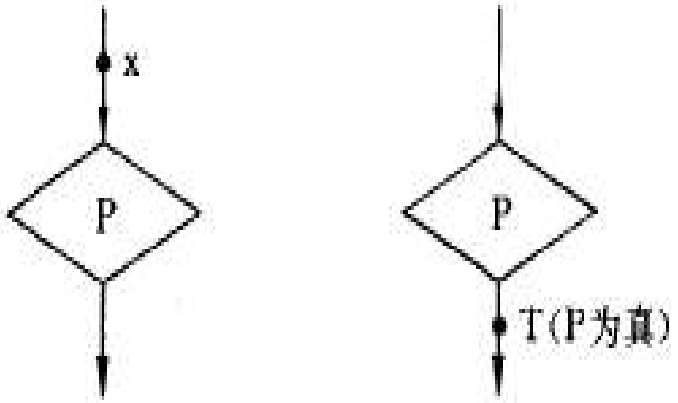
## 运算结点

该类结点可进行算术运算操作和布尔运算操作

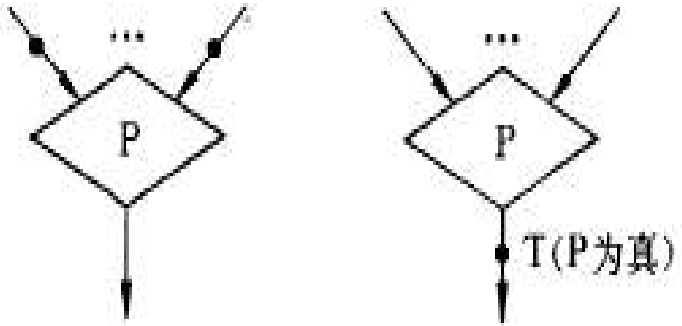



(a) “加” (b) “与” (c) “加1” (d) “非”

## 条件操作结点




单输入分支




多输入分支

# 例：

X1 = 2*A;

D = sqrt(B*B－4*A*C);

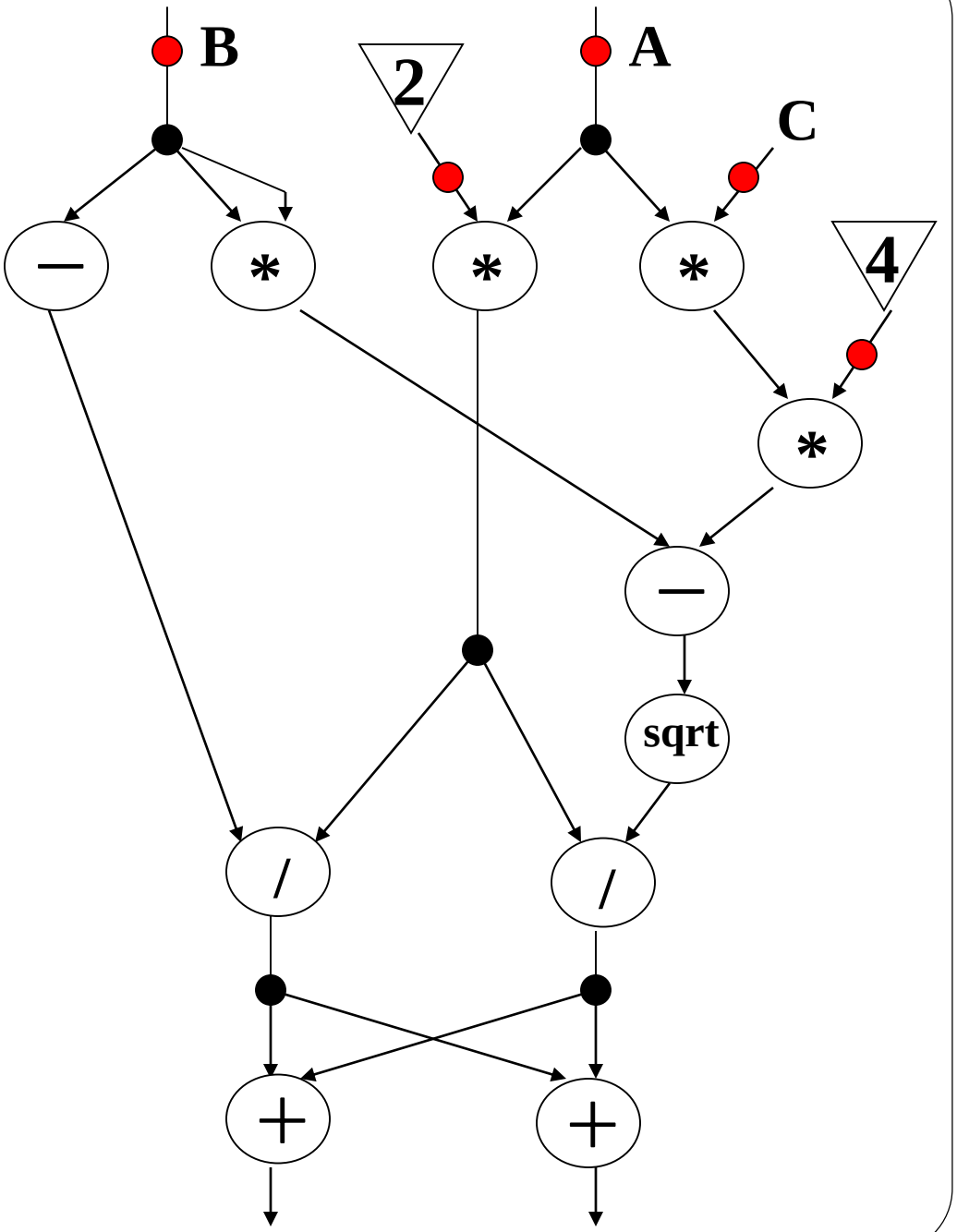
D = D/X1;

X2 = －B/X1;

X1 = X2+D;

X2 = X2－D;

print(X1,X2);



X1 = 2*A;

D = sqrt(B*B－4*A*C);

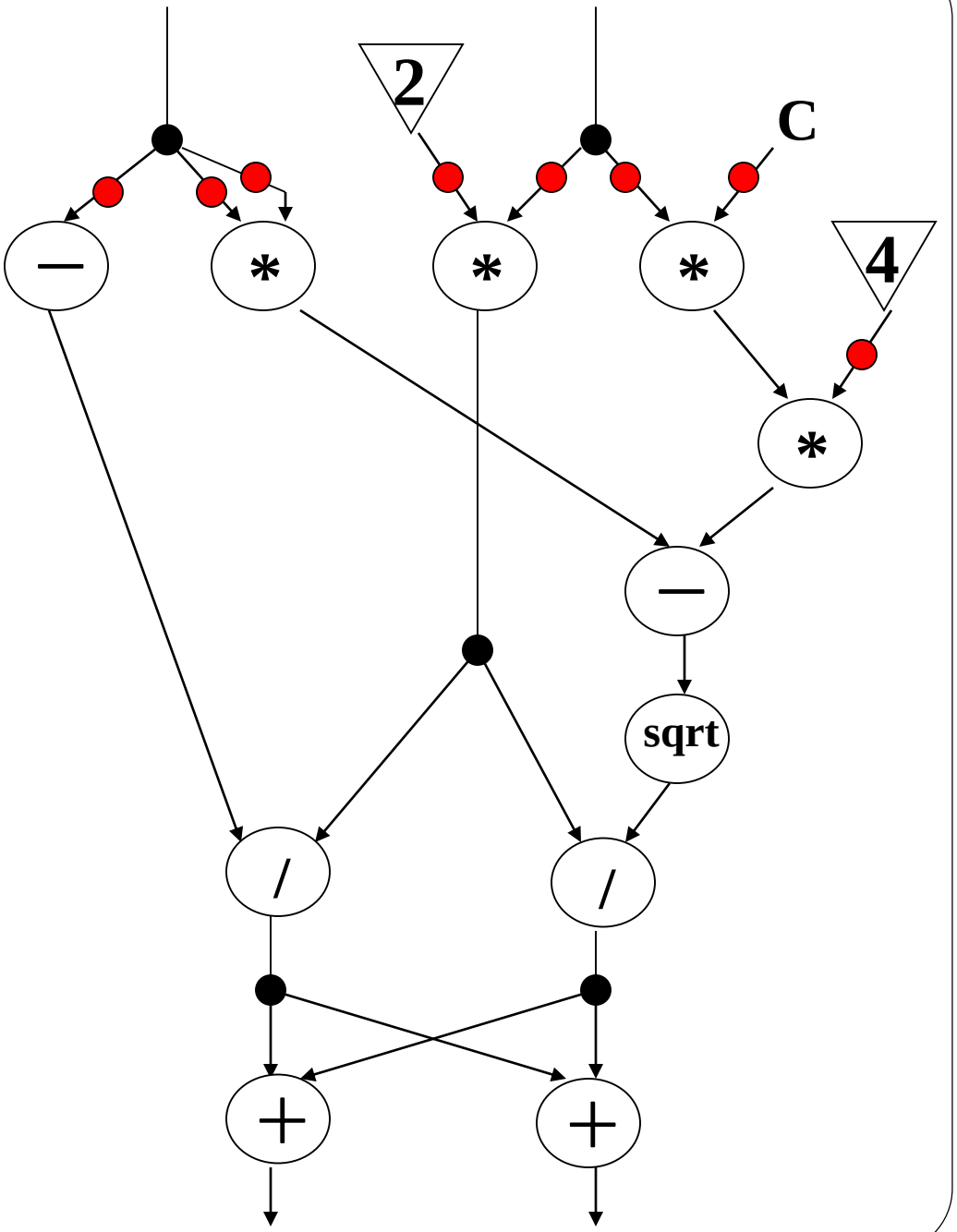
D = D/X1;

X2 = －B/X1;

X1 = X2+D;

X2 = X2－D;

print(X1,X2);



X1 = 2*A;

D = sqrt(B*B－4*A*C);

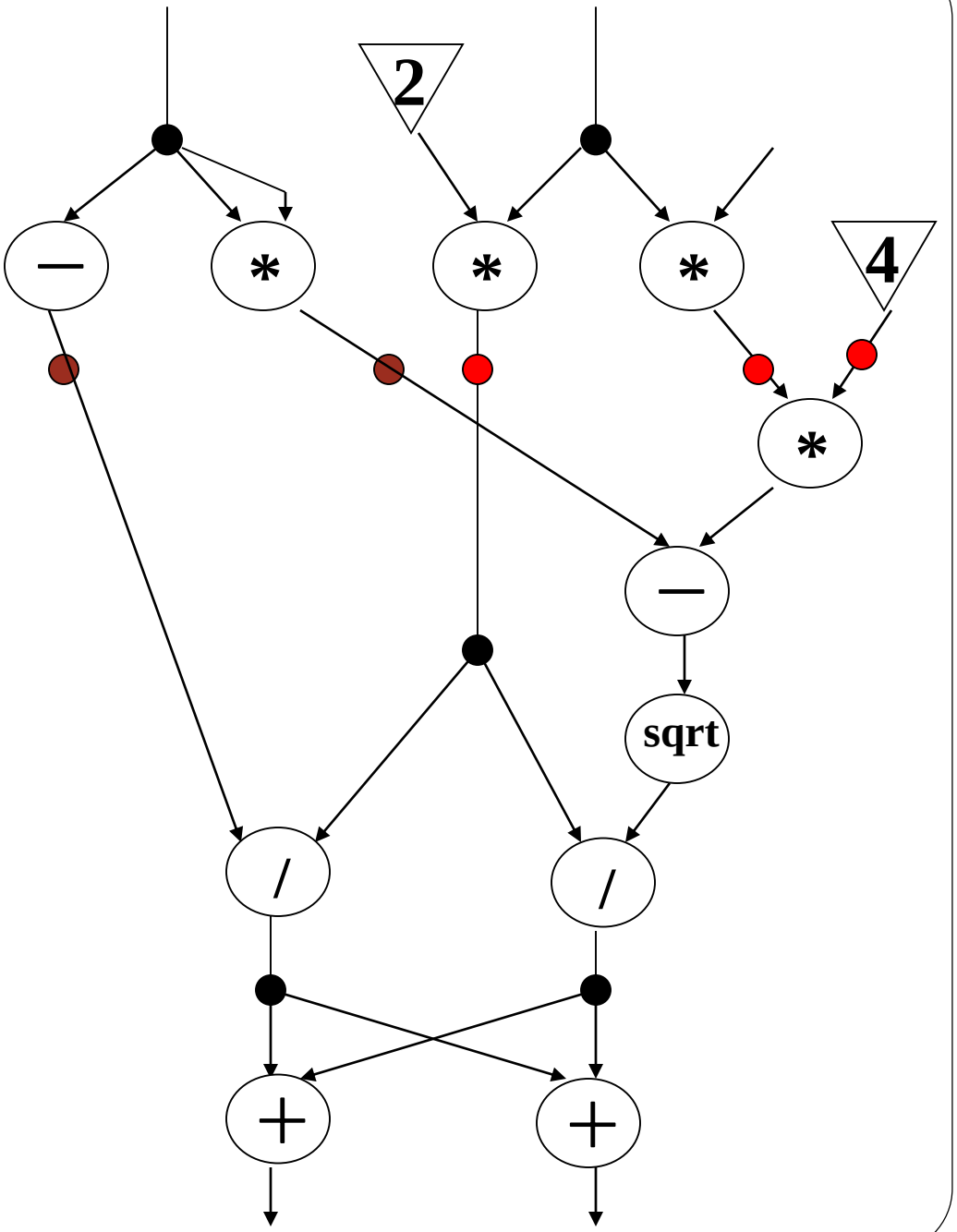
D = D/X1;

X2 = －B/X1;

X1 = X2+D;

X2 = X2－D;

print(X1,X2);



X1 = 2*A;

D = sqrt(B*B－4*A*C);

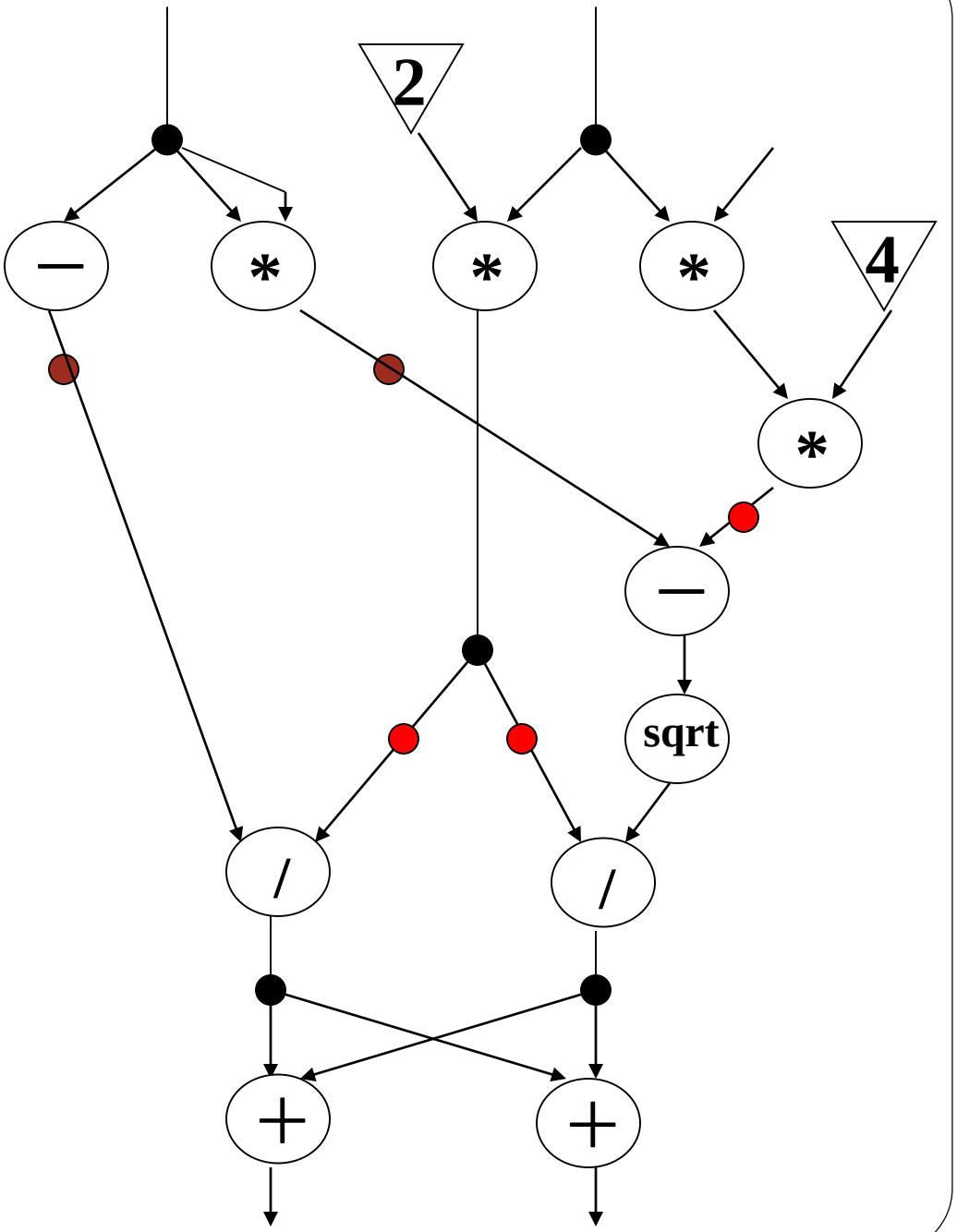
D = D/X1;

X2 = －B/X1;

X1 = X2+D;

X2 = X2－D;

print(X1,X2);



X1 = 2*A;

D = sqrt(B*B－4*A*C);

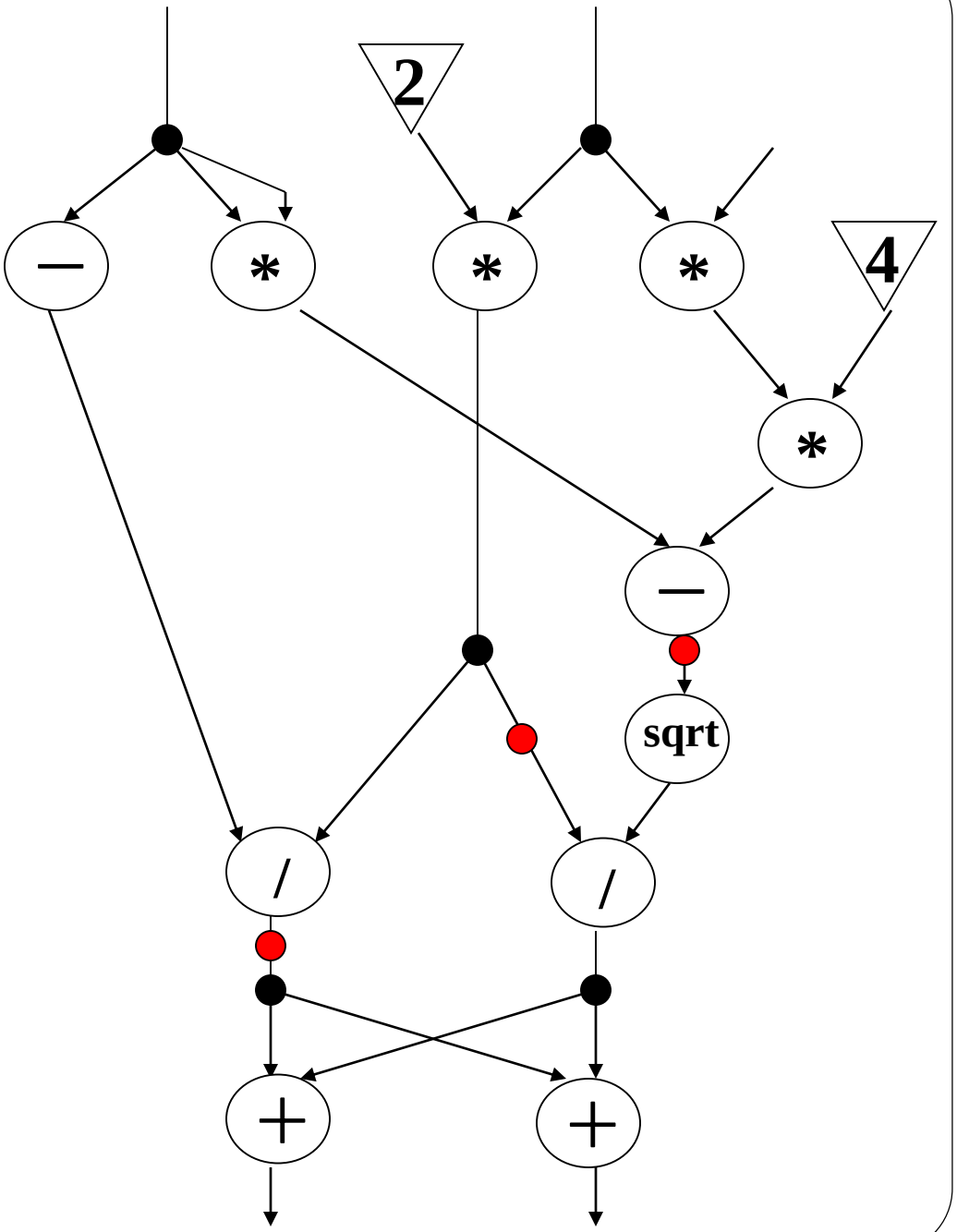
D = D/X1;

X2 = －B/X1;

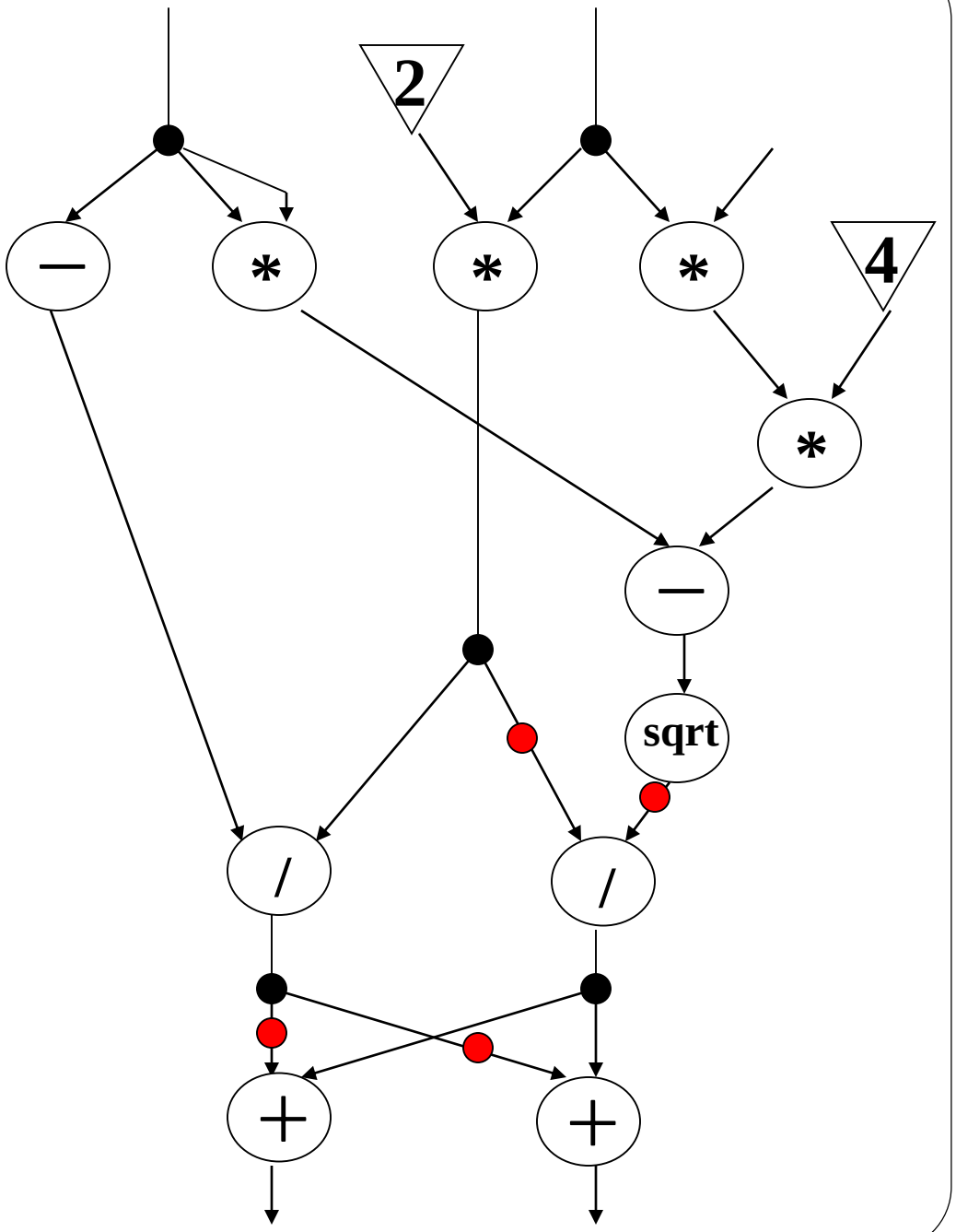
X1 = X2+D;

X2 = X2－D;

print(X1,X2);



```
X1 = 2*A;

D  = sqrt(B*B－4*A*C);

D  = D/X1;

X2 = －B/X1;

X1 = X2+D;

X2 = X2－D;

print(X1,X2);
```



X1 = 2*A;

D = sqrt(B*B－4*A*C);

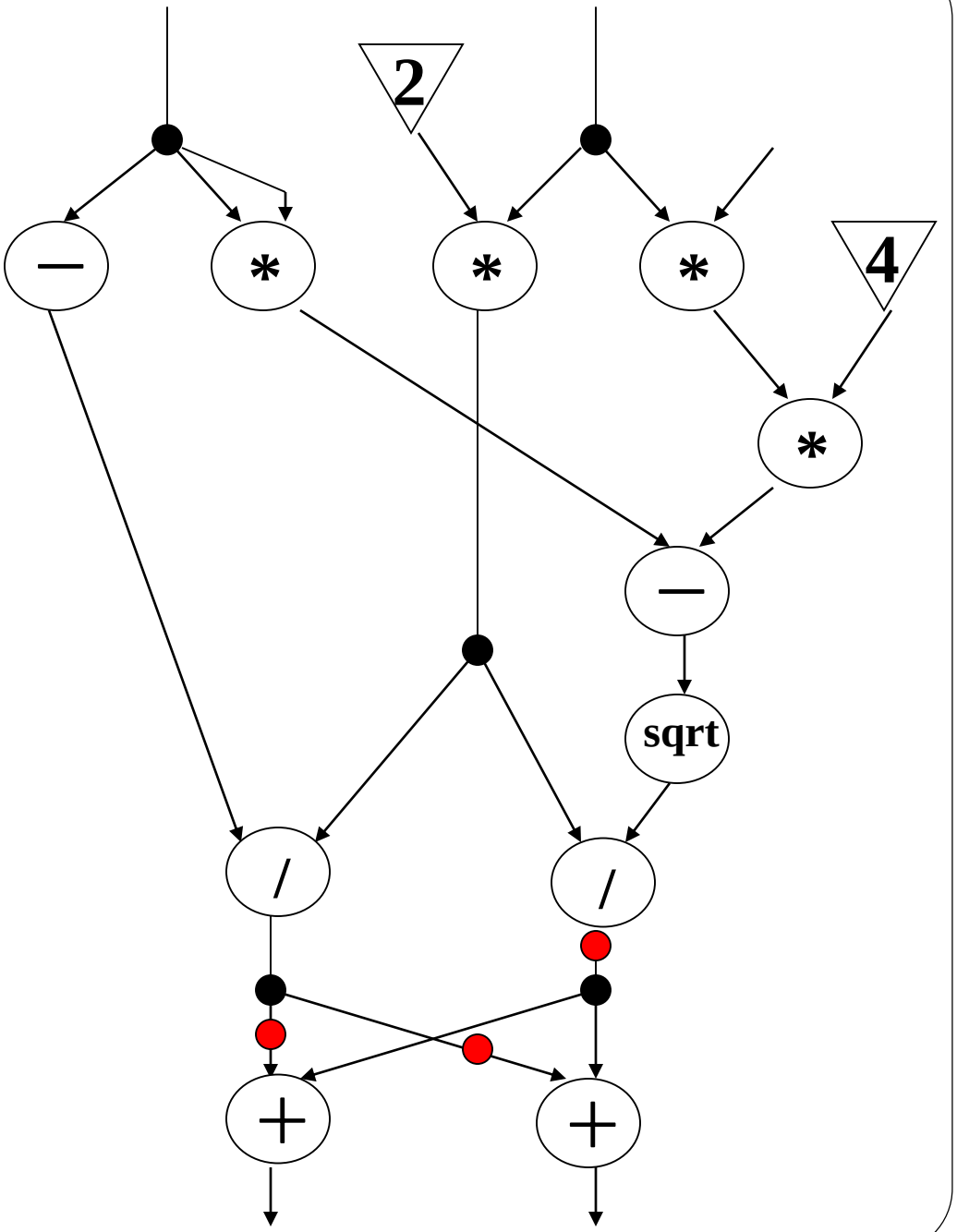
D = D/X1;

X2 = －B/X1;

X1 = X2+D;

X2 = X2－D;

print(X1,X2);



X1 = 2*A;

D = sqrt(B*B－4*A*C);

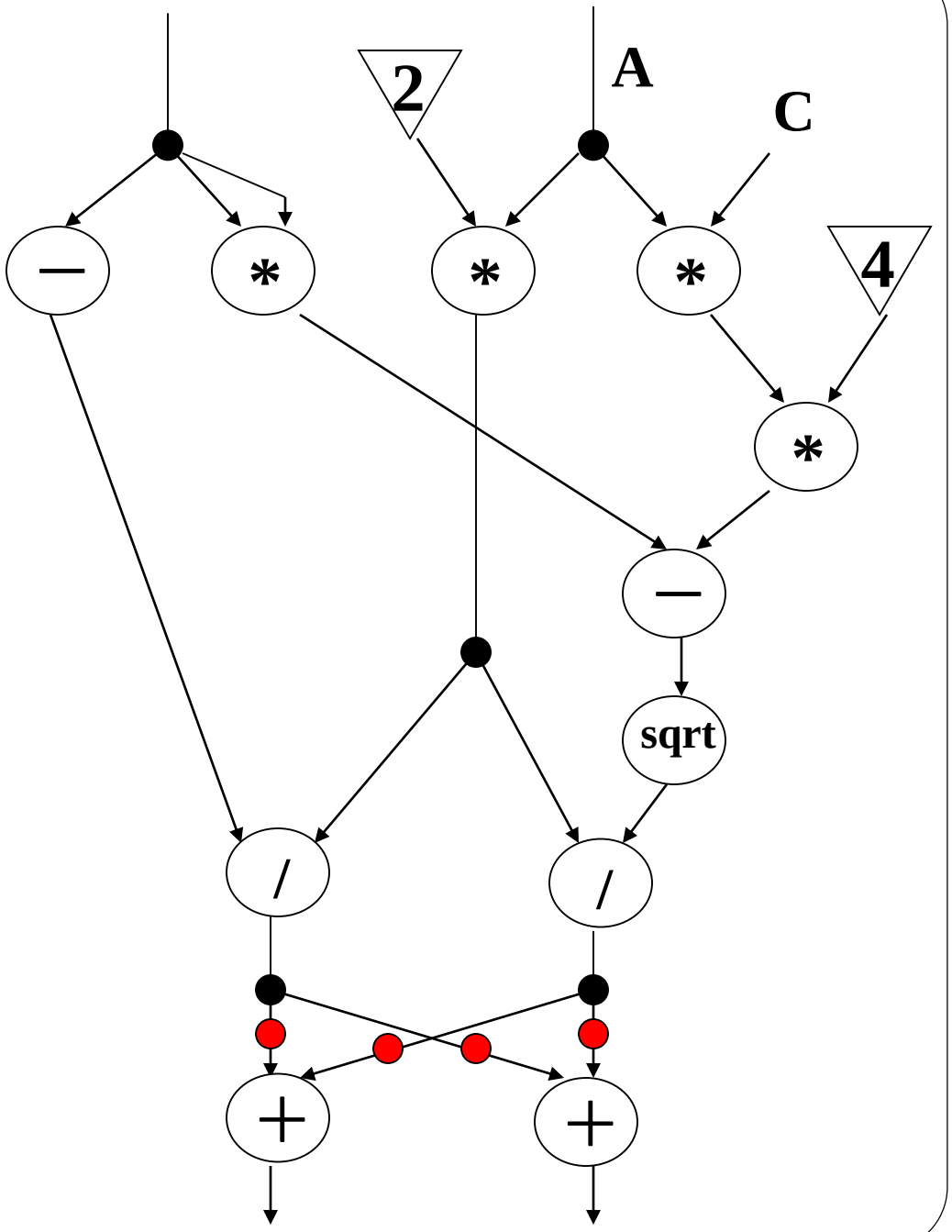
D = D/X1;

X2 = －B/X1;

X1 = X2+D;

X2 = X2－D;

print(X1,X2);



# 顺序结构

X1 = 2*A;

D = sqrt(B*B－4*A*C);

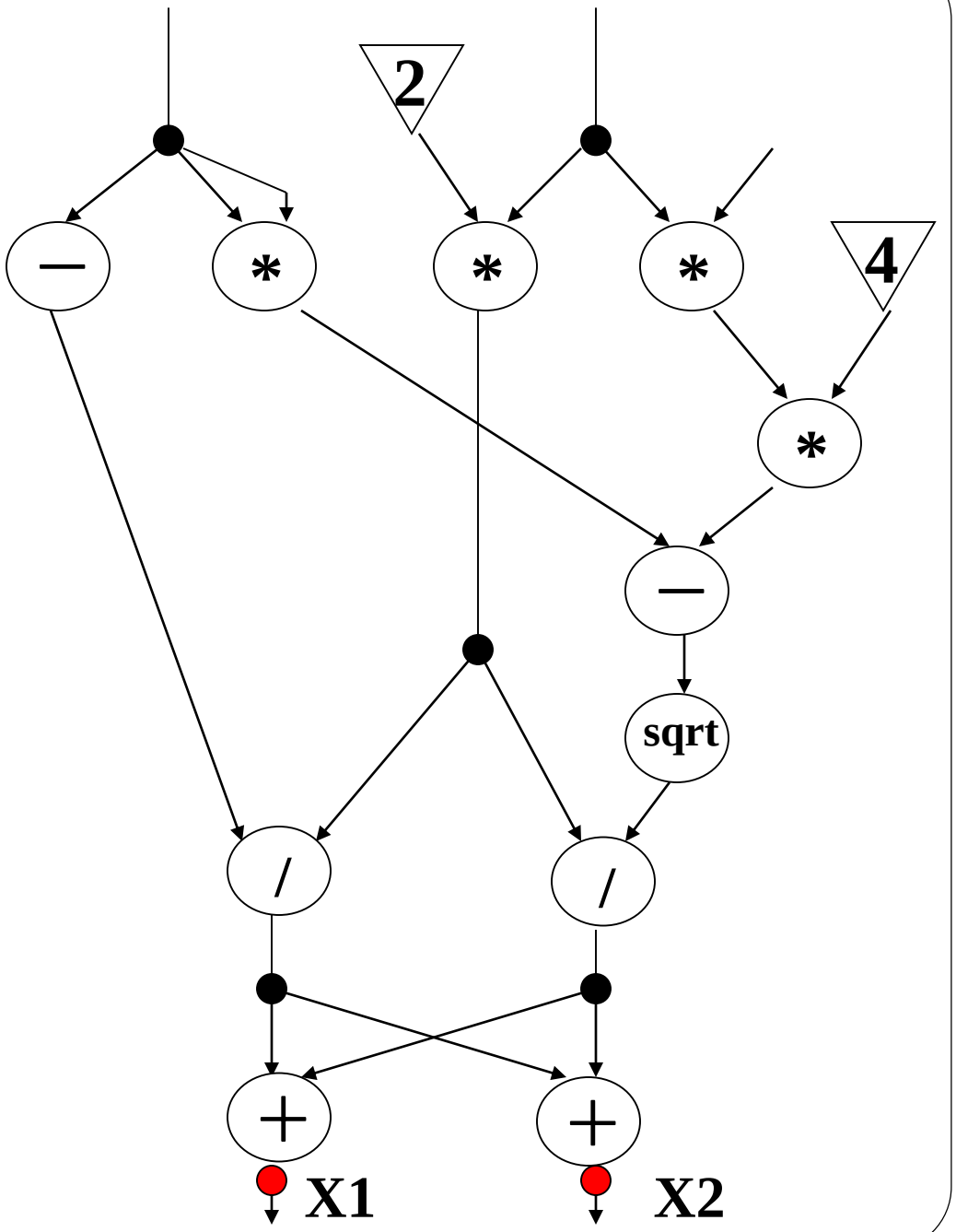
D = D/X1;

X2 = －B/X1;

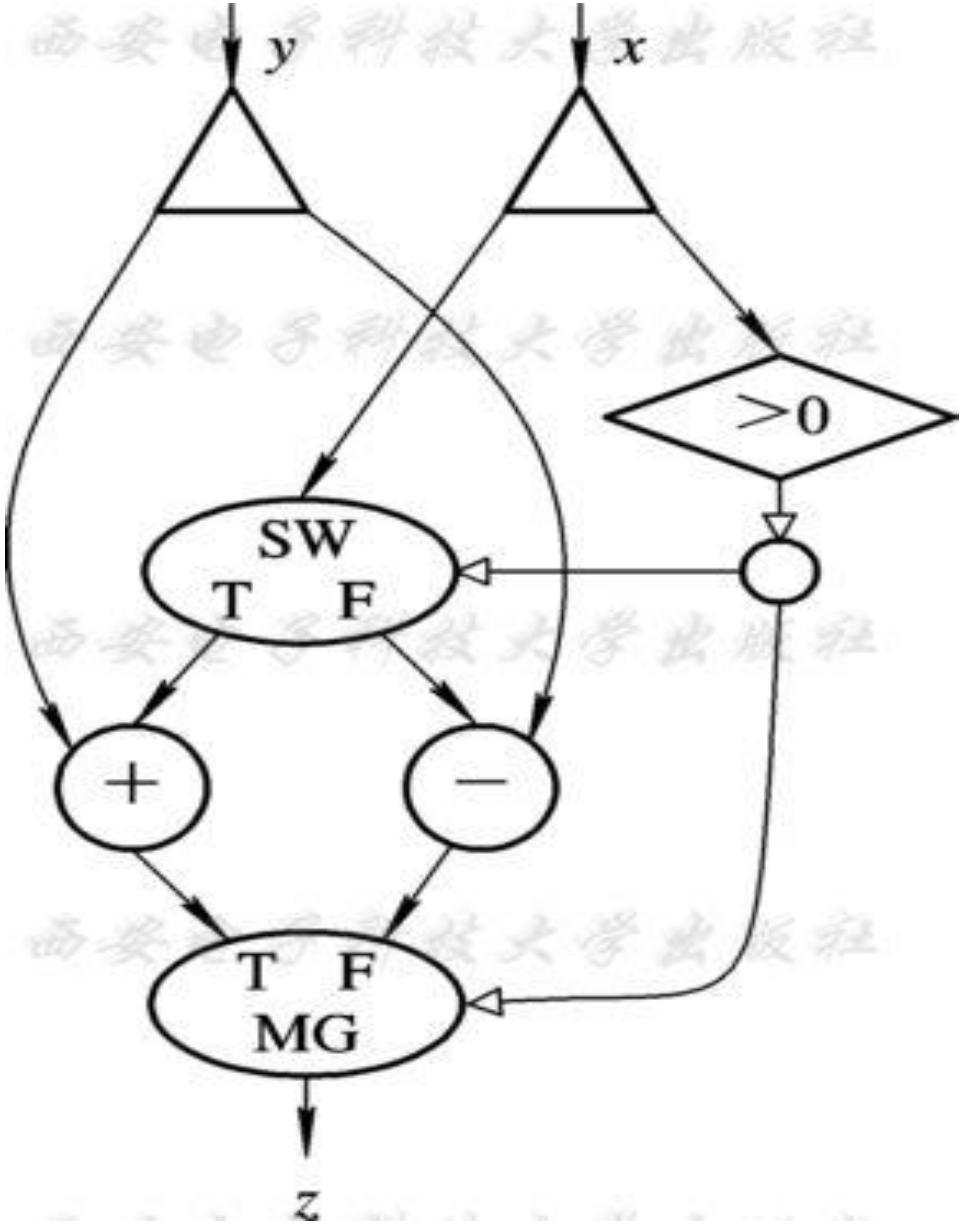
X1 = X2+D;

X2 = X2－D;

print(X1,X2);

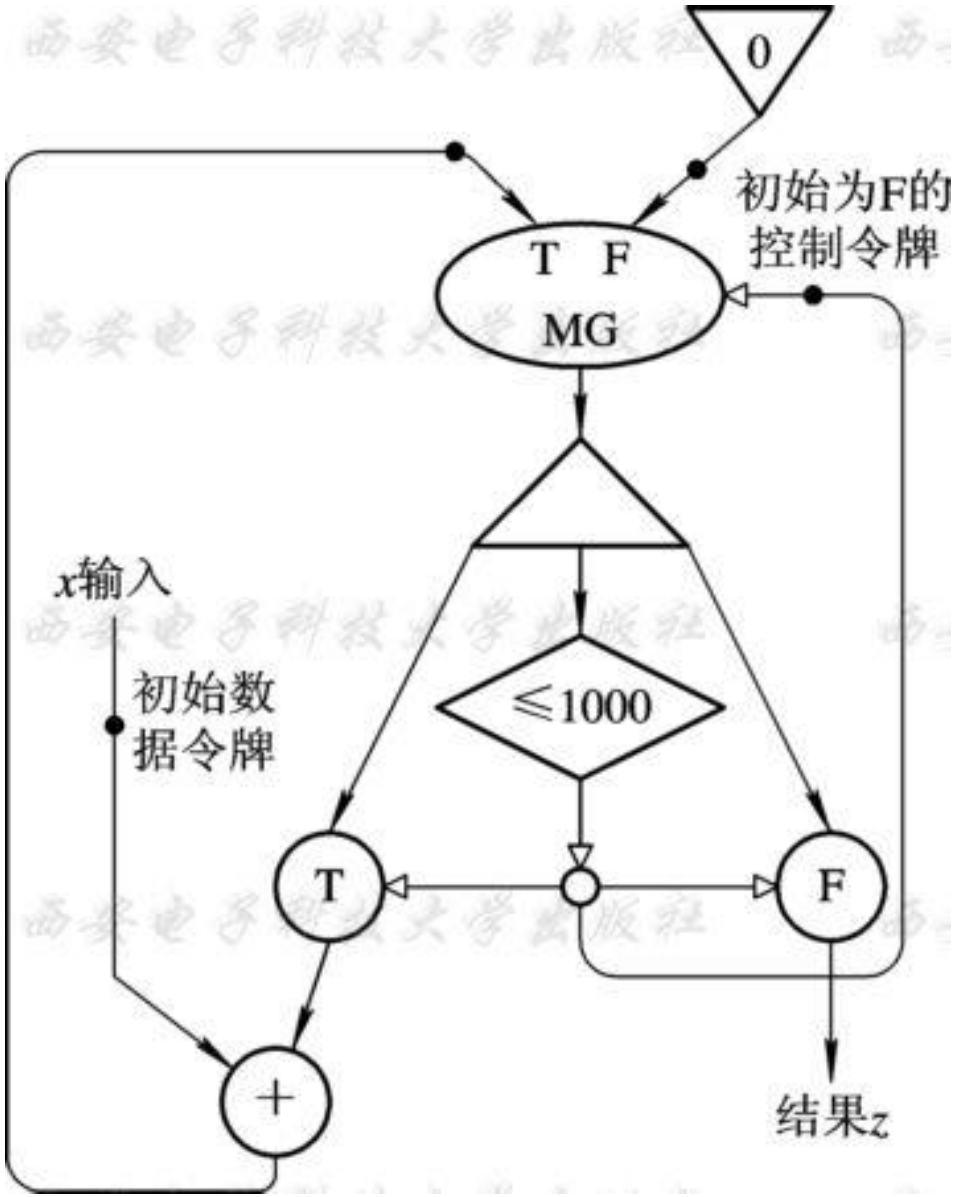


**例：** 实现当$x>0$时，让$x$加$y$，否则，就让$x$减$y$的功能




具有分支结构

**例：实现对x循环累加，直到结果超过1000为止**

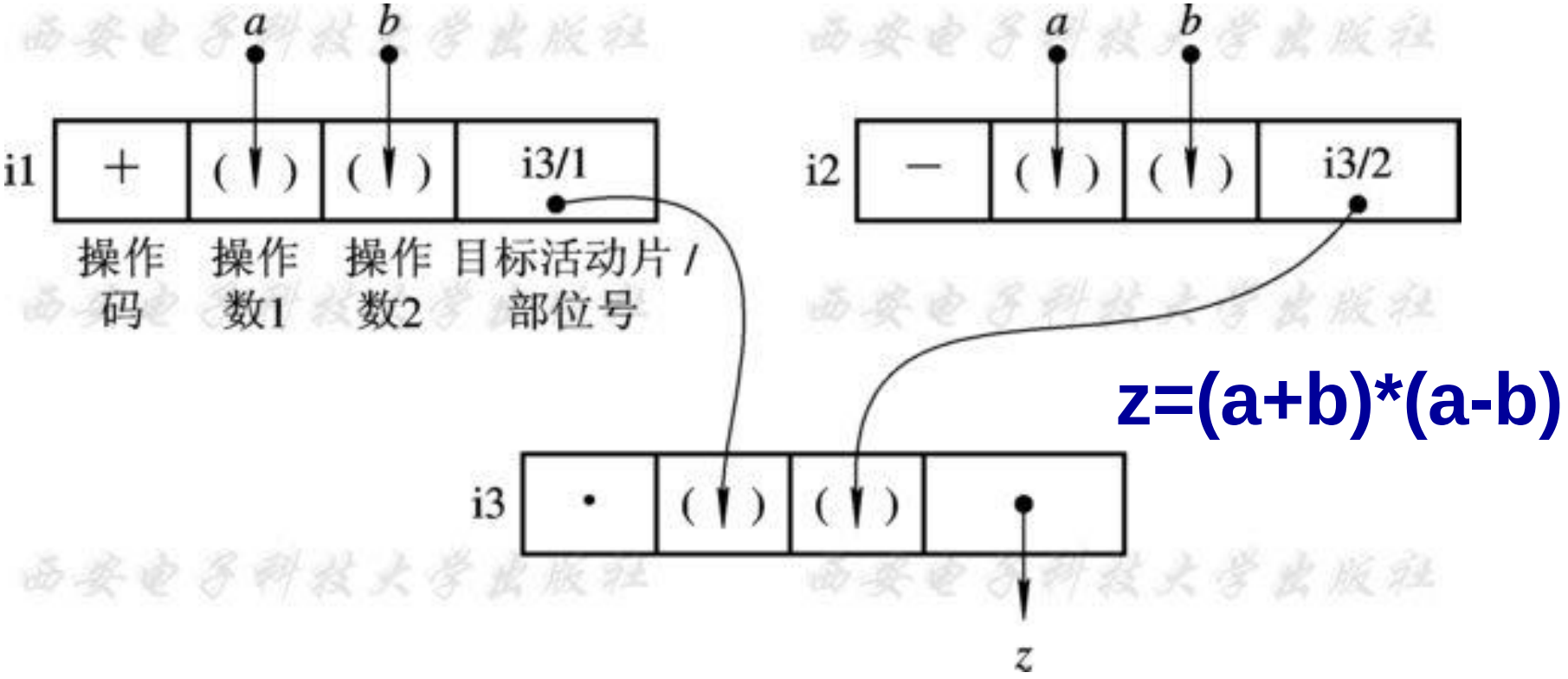



具有循环结构

## 活动模片表示法

一个活动模片相当于有向图表示中的一个或多个操作节点。数据流则是一组活动模片的集合体。

一个活动模片1个操作码域、2个操作数域和1个目的域；其实质相当于是一条指令。



## 2、数据流语言

是编写数据流计算机的程序(即数据流计算机高级语言)

↓

数据流语言(高级语言)

↓

编写程序(适应数据流机)

↓ 由

编译程序进行处理

↓ 变换

数据流程序图 (机器语言)

↓

生成代码

## 数据流语言的特点

**1、并行性好**

指令的执行顺序仅受程序中数据相关性的约束，而与指令在存储器中存放位置无关。

**2、单赋值规则**

在一个程序中，每个变量只赋值一次，不允许同名变量在不同语句的左边出现一次以上，不存在全局变量和局部变量的概念。即为任何一个重新赋值的变量选择另一个从前没有使用过的名字。

程序1：
```
x=a−b
x=x+y
z=x−y
```

程序2：
```
x=a−b
x1=x+y
z=x1−y
```

3、不产生副作用

不使用全局变量和公共变量，严格控制变量使用范围，采用赋值调用，而不是引用调用。

4、结果的局部性

数据流语言完全采用模块化结构，不使用全局变量，不允许全局赋值，对形参的赋值也有严格限制。

5、循环程序迭代展开

动态数据流计算机可以把一个循环程序的不同次循环展开来进行并行计算，从而提高了并行度。

# 6.1.3 数据流计算机的结构

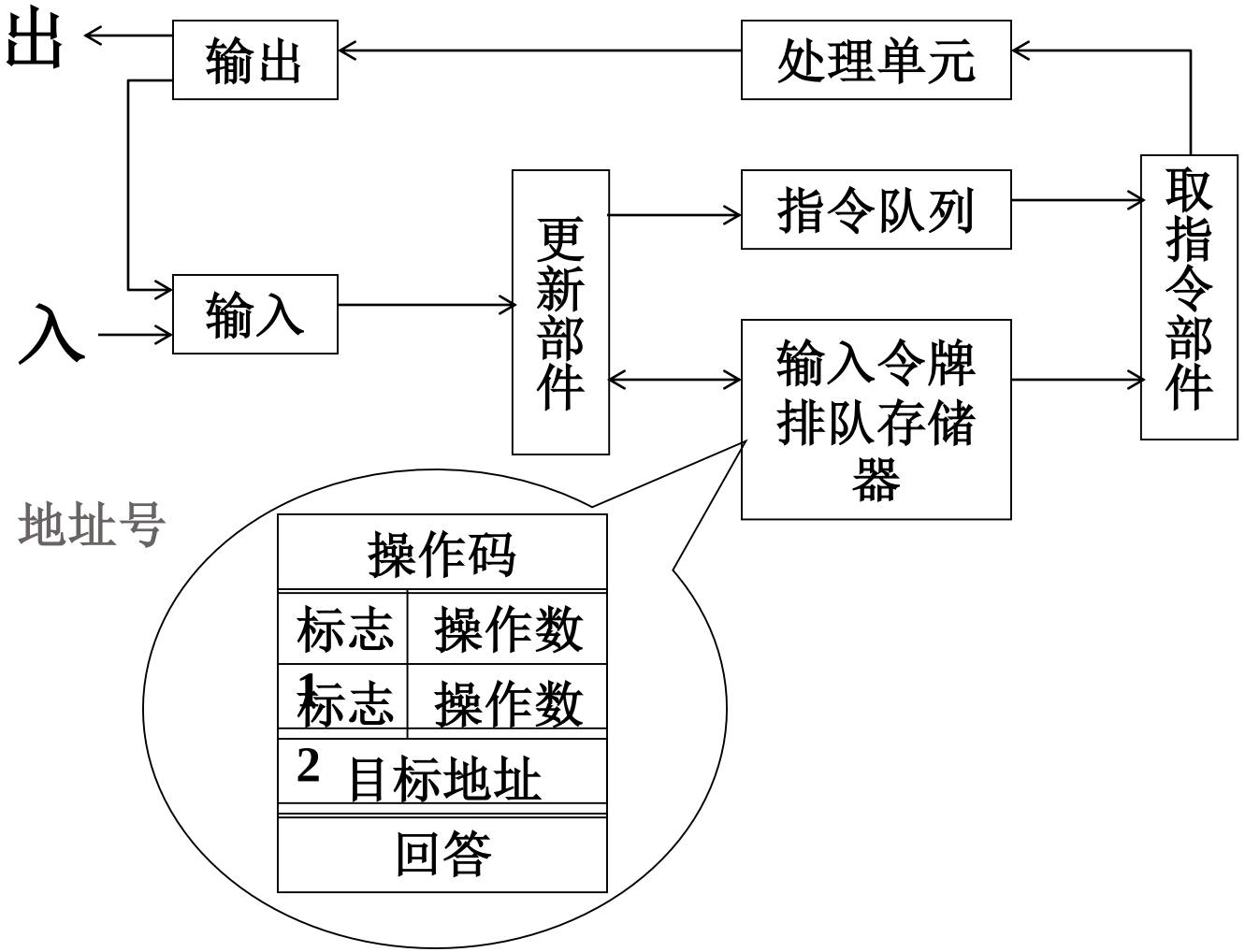
根据对数据令牌的不同处理方式，数据流计算机有两种结构。

*静态数据流计算机:* 任意一个节拍内，任意一条有向分支线上只允许存在一个数据令牌的处理方式, 故数据令牌不加标号.

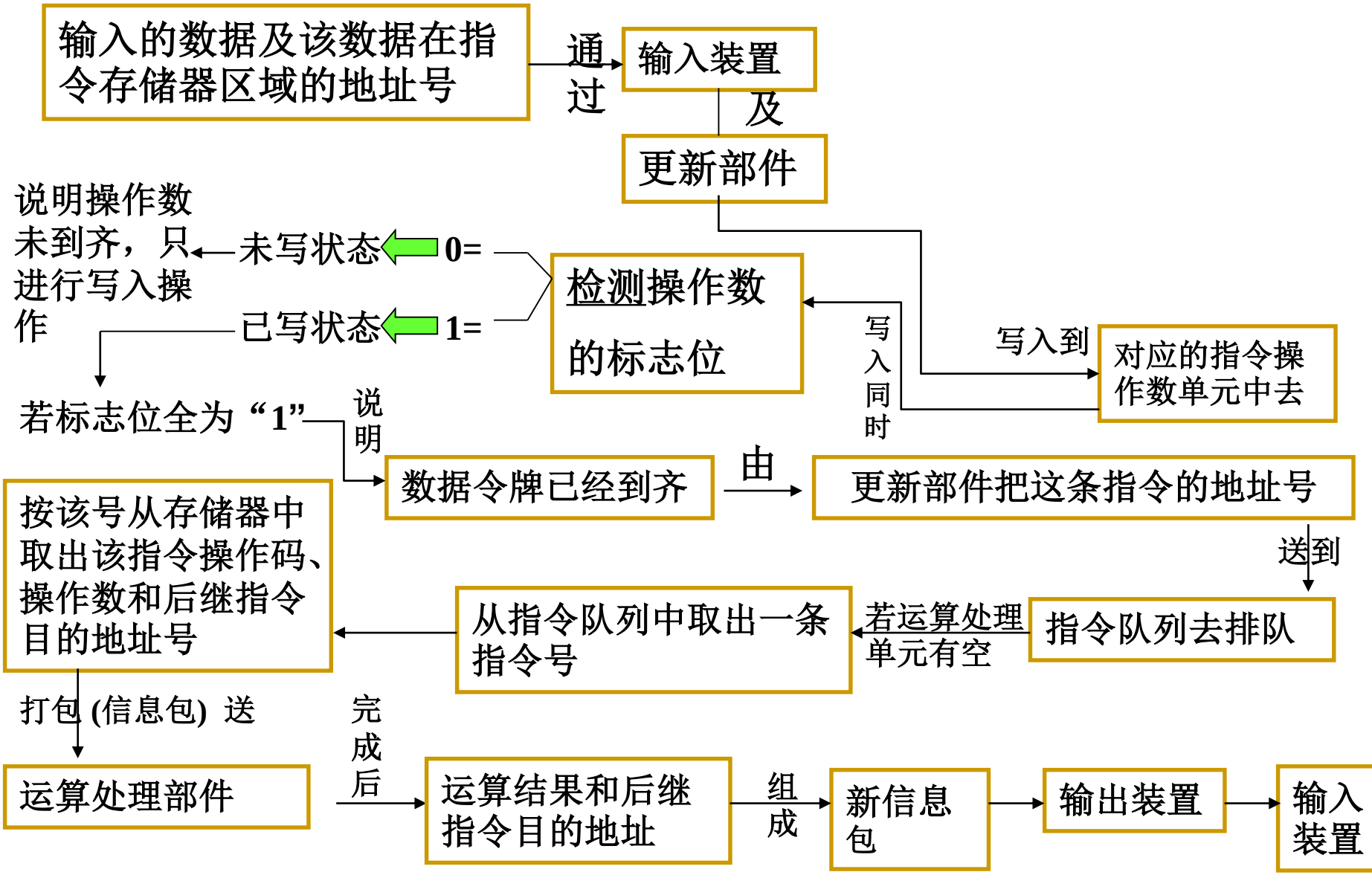
*动态数据流计算机:* 任意一条有向线上可同时传送几个数据令牌,故每 个数据令牌都必须带上标志(令牌标号及其他特征信息).

# ●静态数据流计算机

## 典型结构

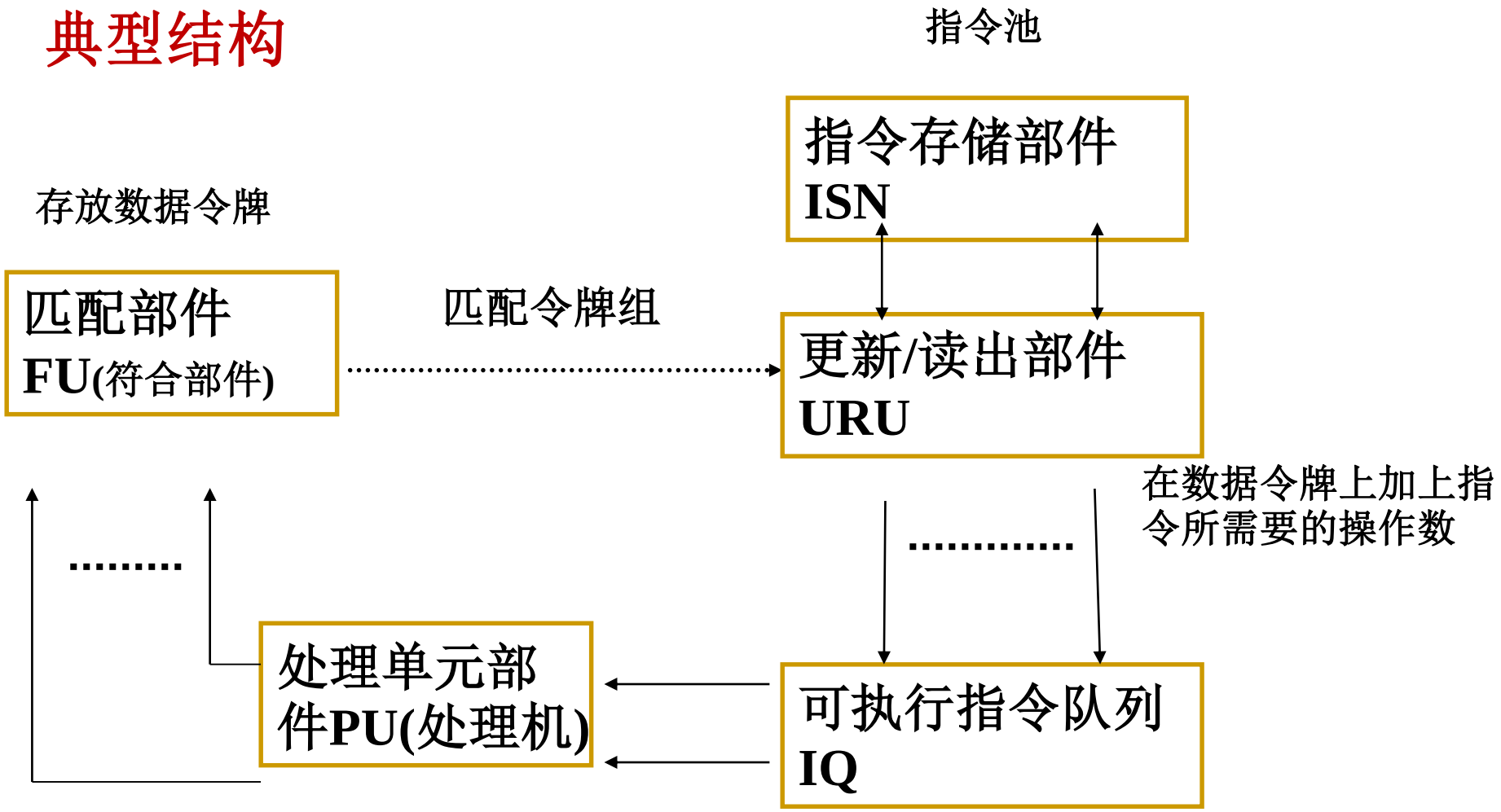


# 工作过程

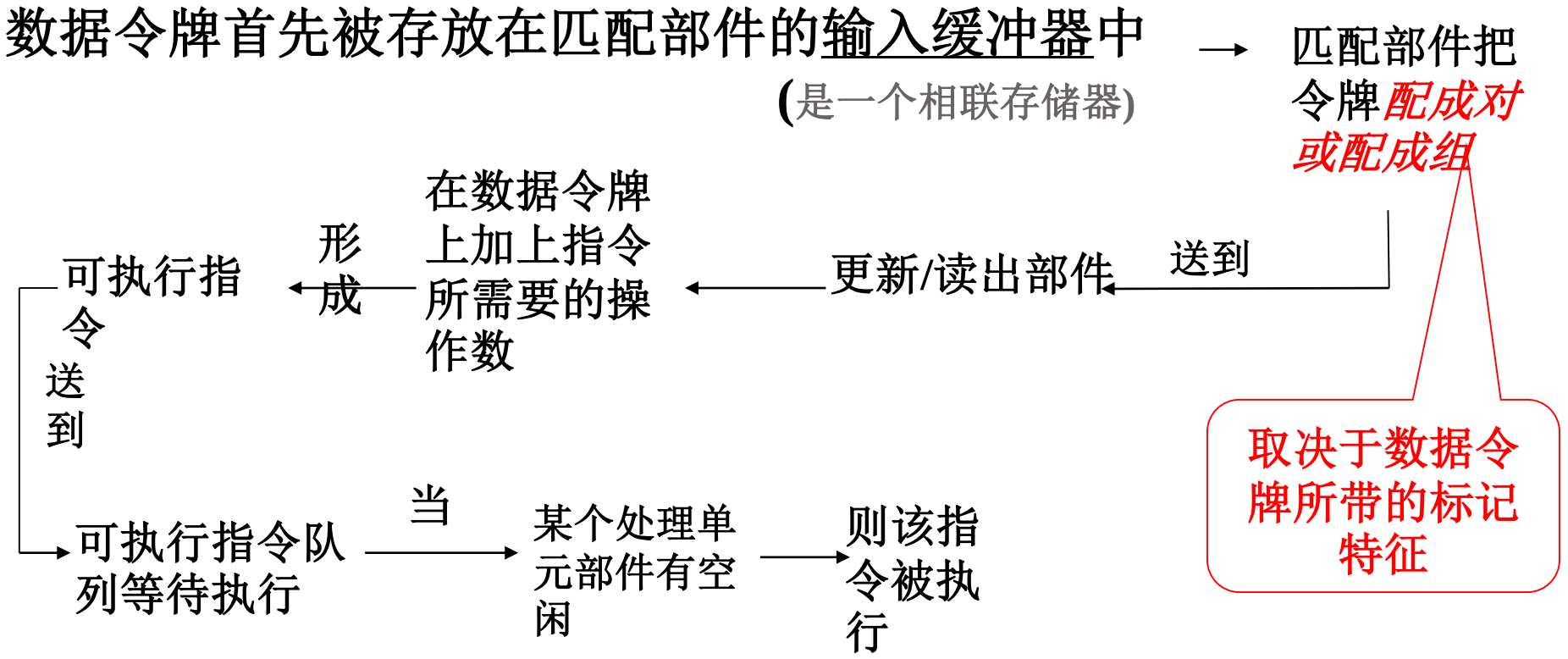

输入的数据及该数据在指令存储器区域的地址号
通过
输入装置
及
更新部件
写入到
对应的指令操作数单元中去
写入同时
检测操作数的标志位
0=
未写状态
说明操作数未到齐，只进行写入操作
1=
已写状态
若标志位全为“1”
说明
数据令牌已经到齐
由
更新部件把这条指令的地址号
送到
指令队列去排队
若运算处理单元有空
从指令队列中取出一条指令号
按该号从存储器中取出该指令操作码、操作数和后继指令目的地址号
打包 (信息包) 送
运算处理部件
完成后
运算结果和后继指令目的地址
组成
新信息包
输出装置
输入装置

# ●动态数据流计算机

## 典型结构



# 工作过程:


数据令牌首先被存放在匹配部件的输入缓冲器中
(是一个相联存储器)
匹配部件把令牌配成对或配成组
送到
更新/读出部件
在数据令牌上加上指令所需要的操作数
形成
可执行指令
送到
可执行指令队列等待执行
当
某个处理单元部件有空闲
则该指令被执行
取决于数据令牌所带的标记特征

# 6.2 归约机

归约机则是需求驱动，执行的操作序列取决于对数据的需求。对数据的需求又来源于函数式程序设计语言对表达式的归约(Reduction)，主要包括：

①所有函数表达式的集合

②所有目标(也是表达式)的集合

③由函数表达式到目标的函数集合

**z=(a+b)*(a-b)**

**z=f(); f() = g()*h(); g() = a+b; h() = a-b**

可见:

◆ 函数是基本成分，是从一批目标到另一批目标的映射。

◆ 从函数程序设计角度看，程序就是一个函数的表达式。

◆ 函数表达式的每一次归约，就是一次函数的应用，或是一个子表达 式(子函数式)的代换(还原)

归约方式体现了按需求驱动的思想，根据对函数求值的需求来激活相应指令

针对函数程序设计语言的特点和问题来设计支持函数式程序运行的新计算机，这就是归约机。

## ●归约机的结构特点：

①归约机应当是面向函数式语言，或以函数式语言为机器语言的非 Neumann型机器

②采用大容量物理存储器和大容量的虚拟存储器，具备高效的动态 存储分配和管理的软硬件支持，以满足归约机对动态存储分配及所需存储空间大的要求。

③处理部分应当是一种有多个处理器(机) 并行的结构形式，以发挥 函数式程序并行处理的特长。

④采用适合于函数式程序运行的多处理器(机)互连的结构 (如树型或多层次复合的结构）形式。

⑤尽量减少进程调度及进程间通信开销,并使各处理机的负荷平衡.

● 归约机的分类——根据机器内部对函数表达式所用的存储方式不同来分

串归约机

◆是一种特殊的符号串处理机，　函数定义、表达式和目标都以 字符串的形式存储于机器中。

◆函数式语言的源程序可以不经 翻译，直接在串归约机上进行处理。

◆串归约机的缺点：不能共享子表达式，多次应用就得多次复制和求值运算，以致时间和空 间的辅助开销较大

图归约机

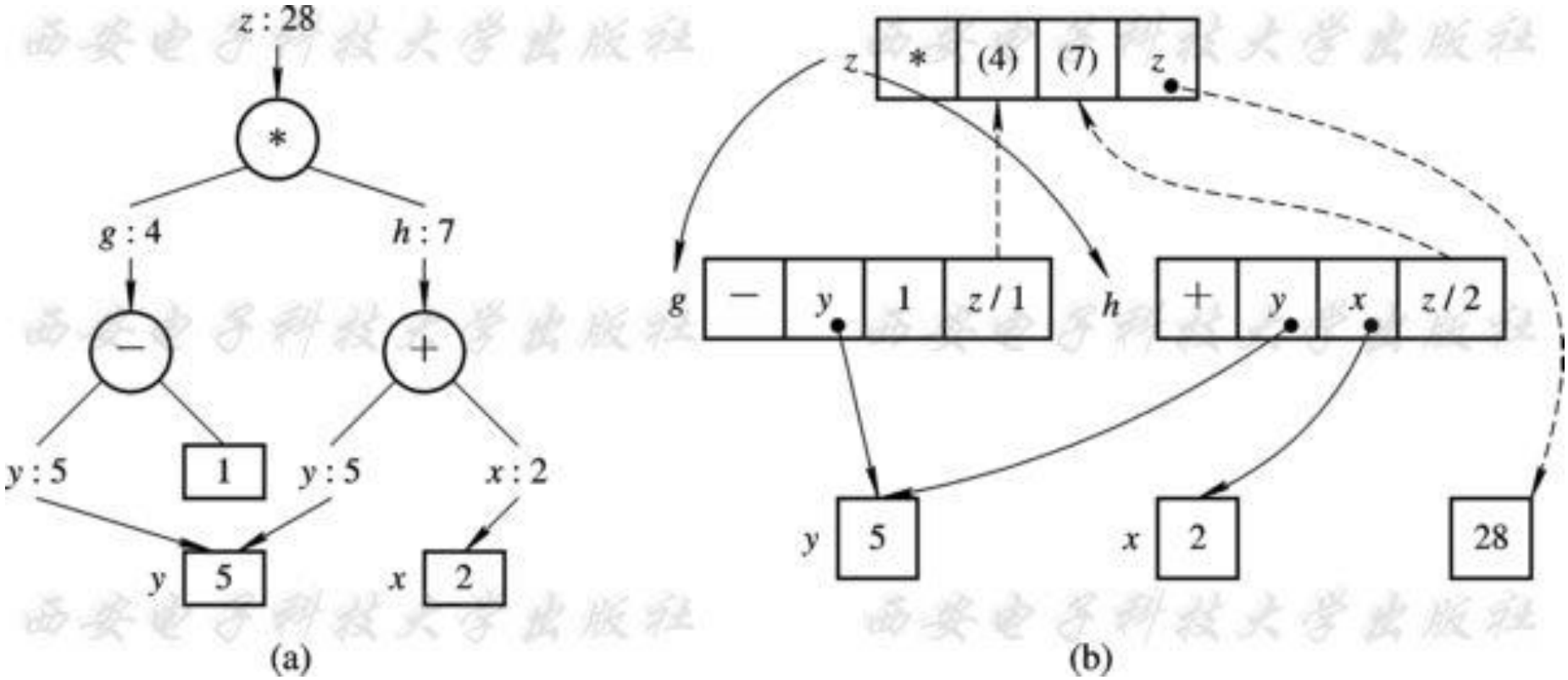
◆采取将函数定义、表达式和目标以图的形式存储于机器中，图是其处理对象。

◆图归约过程免去了频繁的复制　开销，并通过让多个父表达式的指针指向同一子表达式，实现了子表达式的共享，免去了复制。

◆不必强调串归约机要求的邻接 性，使存储空间利用率提高。

◆一旦某结点出错，会使与此结点有关的全部信息丢失，可靠 性不高。

# z=(y-1)*(y+x)




串归约和图归约

**(a)串归约；(b)图归约**

# 练习

1、数据流计算机与控制驱动的控制流的根本不同在于，机器不需要_____程序计数器_____。

2、从语义上，数据流是基于_____异步性_____和_____函数性_____的一种计算模型

3、数据驱动计算机与需求驱动计算机有何不同之处？它们如何描述？是怎么分类的？